2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Navigation on Point-cloud - a Riemannian Metric approach

Ming Liu*, Roland Siegwart
The Hong Kong University of Science and technology
eelium@ust.hk
Autonomous Systems Lab, ETH Zurich, Switzerland
rsiegwart@ethz.ch

***Abstract*—Mobile wheeled- or tracked-robots drive in 2.5-dimensional (2.5D) environments, where the traversable surface can be considered as a 2D-manifold embedded in a three-dimensional (3D) ambient space. In this work, we aim at solving the 2.5D navigation problem solely on point-cloud.**

The proposed method is independent of traditional surface parametrization or reconstruction methods, such as a meshing process, which generally has high computational complexity. Instead, we utilize the output of 3D tensor voting framework (TVF) using raw point-clouds. A novel local Riemannian metric is defined based on the saliency components of TVF, which helps the modeling of the latent traversable surface. Using this metric, we prove that the geodesic in the 3D tensor space leads to rational path-planning results. Compared to traditional methods, the results reveal the advantages of the proposed method in terms of facilitating the robot maneuver with minimum movement.

I. Introduction

The development of robotics is always inspired by human experience and activities. Taking the typical scenario shown in figure 1 for instance: an elder man is trying to fetch the blue cup which is filled with his favorite coffee. Out of rational consideration, he probably would take the cyan (light color) detour rather than the red (dark color) bumpy path. For his situation, it hardly makes sense to take the red path, even though the red path leads to an accumulated shorter distance.

Derived from that, such a concept is usually extended to robotic planning for navigation, using various cost functions depending on selected criteria. The generated path ought to not only regard the shortest Euclidean distance, but also ease the maneuver of the mobile robot.

A. Robotic Challenges

Most existing navigation algorithms for mobile robots take advantage of the assumption that the configuration space of the traversable surroundings is a *developable surface*. It means that the map can be simply considered as a two-dimensional (2D) plane [1], [2] without distortion, or directly projected to a 2D plane without information loss. It is originated from the widely-applied representations derived by mature 2D SLAM techniques [3], such as filtering-based methods [4], [5] or posegraph-based methods [6]. Such an assumption greatly alleviates the requirement on detailed analysis of the terrain. However, the robot ought to deal with the dynamics introduced from the 3D terrain shape in real environments.

This work was supported by HKUST project IGN13EG03, partially supported by the EU FP7 project NIFTi (contract # 247870).

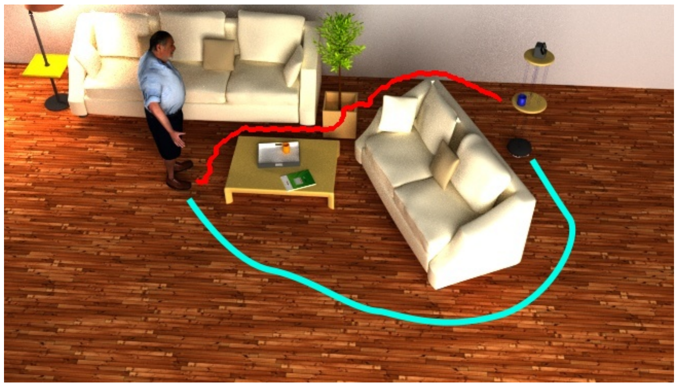
Fig. 1. Typical scenario that implies rational planning

In recent years, the fast development of 3D mapping techniques [7], [8] and sensors [9] makes the modeling of the multi-terrain environment feasible [10] and easy to approach for real applications [11] under task scheduling [12].

As the raw output from these 3D mapping techniques, point-cloud has been widely studied. The following major aspects make the analysis on raw point-cloud a challenging problem, let alone the navigation on it.

- *Unreliability of observation*: the unreliability is multi-fold. Outliers, missing points and non-uniform distribution of points are the major drawbacks of point-cloud.
- *Large amount of sparse data*: the 3D sensor usually generates a large amount of points per scan. However, points are not continuously defined in the 3D space. The missing information among points, especially latent structural information, must be recovered by subtle filters.
- *Computational complexity*: due to the large amount of data, the high computational cost is a bottleneck.

Because of these challenges, many works try to avoid the direct operations on points, leading to various representations, e.g. meshed surface [13] or tree based structures [14].

Despite these difficulties, we tackle the surface modeling problem using raw point-cloud as input in this paper. It is because we aim at solving the related problems in real-time, and the raw point-cloud is the most readily available representation. In order to cope with the large amount of sparse data, we adopt tensor voting framework (TVF) [15] for this study. Respecting with the computational complexity, we utilize a GPU implementation [16] proposed by us, which enables the TVF calculation in near real-time. After that, we construct a novel local Riemannian metric based on saliency components of the TVF, which aids the further analysis of the surface properties and the corresponding tensor field.

As a typical application, we present how such metric can be used for trajectory planning, for which a functional property derived from TVF has been adopted. The *stick saliency* of TVF indicates the strength of local planarity. This information is used to help generating a trajectory that can be viewed as a "rational path", i.e. try to drive along flat areas, avoid climbing redundant slops and avoid paths that lead to too much vibration on the way. The mathematical criteria for the evaluation are introduced in section V.C.

II. Related Work

A. Tensor Voting Framework (TVF)

Tensor voting [15] is originated in computer vision. It has been applied to several applications, such as segmentation [17], [18]. Through these works, tensor voting has shown its importance in reconstructing missing structures and local information registration [19], [20]. We consider it as one of the most important algorithms for structural analysis, because it outperforms other methods by its tolerance to noise and missing data, its consistency for local information and intuitive extraction of evidence saliency etc. Preliminary work using TVF for planning has been proposed using a iterative algorithm over the dense voting grids [21], where the generated path adapts to smooth local curvature due to the non-holonomic motion constraints. In this paper, we use the results from sparse voting directly and consider not only the shortest distance, also to facilitate the robot maneuver. The local information is represented by a Riemannian metric.

Nevertheless, the computational cost of tensor voting is high. The original algorithm has complexity $O(N^2)$, where N is the number of points in the point-cloud. In our previous work, we proposed a parallel computation, which was further optimized considering the advanced calculation characteristics of CUDA, in order to improve calculation efficiency, e.g. using coalesced memory access, avoiding atomic operation and implementing online tensor split etc[16].

B. Geodesic on Point-cloud

Provided the surface model, the geodesic calculation is relatively easy, such as using iterative midpoint search or approximating by descent gradient [22]. An efficient method based on contour propagation was proposed in [23]. Meanwhile, several works have put effort to calculate geodesic on point-cloud. Ruggeri et al. proposed an approximation method based on energy minimization, regarding errors in local surface fitting [24]. In our case, the local surface information is embedded in TVF, which is computationally more efficient since the voting procedure is linear in time on GPU. Mémoli and Sapiro explain the intrinsic properties for geodesic on point clouds, considering the case of manifold sampling [25].

However, all these existing works deal with the point-cloud only in Euclidean space, i.e. $\mathbb{E}^3$, which equips a Riemannian metric as a rank-three identity matrix. It introduces inherent drawbacks for robotic navigation, since it assumes the whole configuration space is linked by straight geometry. In order to ameliorate that, we define a Riemannian metric in the tensor space $\mathbb{T}^3$ introduced by TVF. The smoothness of such a space indicates the similarity of local smooth structures instead of direct distances. By analysing the metric of TVF ($\mathbb{T}^3$) in terms of the direction of eigen vectors, we obtain the weights for edges of a graph constructed by sample points.

III. TVF and sparse voting

Tensor Voting [15] is a computational framework used for structural extraction based on saliency of basic evidences. It originated in computer vision problems. King extended its application regarding point-cloud based terrain modeling and proposed an optimized stick voting field [20]. Following a generic pipeline described in [26], we construct sparse ball voting fields $eyes(3)$, and broadcast it through each neighboring point by a exponential decay function. For robotics applications, the kernel size can be chosen as the size of the navigation footprint. In this work, we omit *dense voting* process, which is often performed after sparse voting. Nevertheless it is a powerful tool for further structural inference when needed.

The collected votes from each point lead to a tensor containing the neighbouring structural information. The eigen decomposition of the 3×3 tensor T can be formulated as:

$$\begin{aligned} T &= \alpha_1 \hat{e}_1 \hat{e}_1^T + \alpha_2 \hat{e}_2 \hat{e}_2^T + \alpha_3 \hat{e}_3 \hat{e}_3^T \\ &= (\alpha_1 - \alpha_2)\hat{e}_1 \hat{e}_1^T + && \text{(stick component)} \\ &(\alpha_2 - \alpha_3)(\hat{e}_1 \hat{e}_1^T + \hat{e}_2 \hat{e}_2^T) + && \text{(plate component)} \\ &\alpha_3(\hat{e}_1 \hat{e}_1^T + \hat{e}_2 \hat{e}_2^T + \hat{e}_3 \hat{e}_3^T) && \text{(ball component)} \end{aligned} \tag{1}$$

where α_i's are eigenvalues sorted in decreasing length sequence, $\hat{e}_i$'s are the corresponding eigenvectors. The stick saliency can be represented by $\lambda_1 = \alpha_1 - \alpha_2$. The stick saliency for each point indicates how confident that a point can be considered as lying on a local plane. The corresponding tensor, indicating the plane, is characterized by the normal direction of the local plane $\hat{e}_1$.

IV. Riemannian Metric equipped on TVF

A. Projection and transition function

The embedded manifold M can be interpreted as shown in figure 2. For each point s in the point-cloud, subjecting to the local latent surface, a neighborhood homeomorphic ψ^{-1} to an open subset in the TVF space. The corresponding manifold is equipped with a Riemannian metric g, which determines the inherent relations between two projected points. Adopting the results from TVF, we build the metric g such that local *roughness* of the latent surface can be reflected. This property is specifically interesting for the use-case of trajectory planning. Usually, the optimal trajectory is not necessarily the shortest; instead, the smoothness in curvature is also important. For example, if the one with the shortest Euclidean distance passes through rough terrain or induces complex morphological adaptations, a relatively longer but *flatter* trajectory is preferable, considering risks and power consumption for the robot.

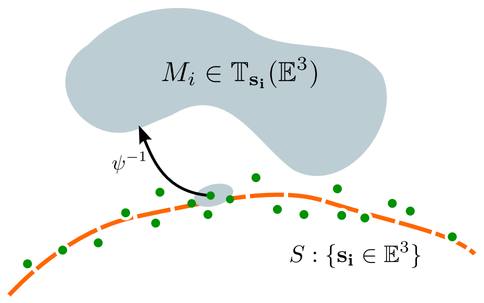
$M_i \in \mathbb{T}_{\mathbf{s_i}}(\mathbb{E}^3)$
ψ^{-1}
$S : \{\mathbf{s_i} \in \mathbb{E}^3\}$

Fig. 2. Intuition of the embedded local manifold for points in $\mathbb{E}^3$. The dashed line indicates the latent surface, where the points lie on.

B. Metric definition

The Riemannian metric on M is a family of inner products on each tangent space T_pM, such that it depends smoothly on p, $\forall p \in M$. However, the specification of the T_pM is trivial if and only if it possesses a frame of global sections, e.g. a vector field is defined on M. [1] Therefore, for the case that a tensor field has already expanded the point-cloud S, the Riemannian metric g can be arbitrarily defined, as long as it is canonical.

Regarding the components introduced in TVF, we see that the stick component indicate the flatness of a local latent surface, with eigenvector expanding $\hat{\boldsymbol{e}}_{\boldsymbol{1}}^T\hat{\boldsymbol{e}}_{\boldsymbol{1}}$; on the other hand, the orthogonal plane is expanded by $\hat{\boldsymbol{e}}_{\boldsymbol{2}}^T\hat{\boldsymbol{e}}_{\boldsymbol{2}} + \hat{\boldsymbol{e}}_{\boldsymbol{3}}^T\hat{\boldsymbol{e}}_{\boldsymbol{3}}$. Inspired by the 2D metric introduced in [28] and [29], with symmetric positive-definite tensors, we propose the following metric:

$$\begin{aligned} g(s) := \mathcal{I}_3 &+ \phi\big(\lambda_1(s)\big)\,\big(\hat{\boldsymbol{e}}_{\boldsymbol{1}}(s)\hat{\boldsymbol{e}}_{\boldsymbol{1}}(s)^T\big) \\ &+\psi\big(\lambda_1(s)\big)\,\big(\hat{\boldsymbol{e}}_{\boldsymbol{1}}(s)\hat{\boldsymbol{e}}_{\boldsymbol{1}}(s)^T + \hat{\boldsymbol{e}}_{\boldsymbol{2}}(s)\hat{\boldsymbol{e}}_{\boldsymbol{2}}(s)^T\big) \end{aligned} \tag{2}$$

where $\mathcal{I}_3$ is identity matrix; $\lambda_1(s)$ is the normalized stick saliency and $\hat{\boldsymbol{e}}_{\boldsymbol{i}}(s)$ indicates the i-th component of the local tensor for site s derived from TVF. Using the proposed metric, we intend to guarantee the following two properties:

- A unit vector is along the most desirable curve when the vector is normal to $\hat{\boldsymbol{e}}_{\boldsymbol{1}}(s)$;
- A unit vector is along the most undesirable curve when the vector is parallel to $\hat{\boldsymbol{e}}_{\boldsymbol{1}}(s)$.

For example, if the embedded vector is the control velocity to the robot, it means that we want to keep the robot move on flat surface as much as possible; at the same time, avoiding climbing redundant hills etc. Guided by these properties, we define the co-efficiencies $\phi(\cdot)$ and $\psi(\cdot)$ using the combination of exponential functions, where we need to ensure $\frac{\phi(x)}{\psi(x)}$ is monotonically increasing and $\frac{\phi(0)}{\psi(0)} = 1$. We set

$$\phi(x) = \frac{e^{kx}}{e^{kx} + e^{-kx}} \text{ and } \psi(x) = \frac{e^{-kx}}{e^{kx} + e^{-kx}} \tag{3}$$

Please notice that the ball saliency is not considered, since the free points are less of interest for structural information.

[1]The proof for this proposition is omitted here. The readers are referred to [27] for details.

C. Distance derivation

Using the proposed metric g, the Riemannian curve length $\mathcal{C}$ is obtained by the integral on M, as shown in equation (4), where τ_s defines the direction derivatives on the manifold. Given the results in (4), the following properties can be drawn:

- The length decreases when the direction of $\hat{\boldsymbol{e}}_{\boldsymbol{1}}$ diverges from τ_s, where $|\tau_s^T\hat{\boldsymbol{e}}_{\boldsymbol{1}}|^2$ indicates the cosine of the separation angle.
- When the stick saliency λ_1 grows, the length decreases, which coincidently means that the integral path lies on a *flatter* surface.
- The local minimal direction is orthogonal to the secondary eigen vector $\hat{\boldsymbol{e}}_{\boldsymbol{2}}$, where $|\tau_s^T\hat{\boldsymbol{e}}_{\boldsymbol{2}}|^2$ is minimized. It implies that the local geodesic is along the $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$ direction (or its opposite) in $\mathbb{T}_s$. This information may help the exploration task for mobile robots, since the corresponding direction is the easiest one to approach. Further discussion is not included in this paper.
- Based on the last item, when the robot is moving in the direction of $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$, the Euclidean distance is used. Otherwise, the curve length is greater than then Euclidean distance.
- It does not depend on parameterisation of the curve, i.e.

$$|\mathcal{C}|_{\mathcal{M}} = \int_{\mathcal{C}} \sqrt{\tau_s^T g(s)\tau_s}\ ds = \int_{\mathcal{C}'} \sqrt{\tau_{s'}^T g(s')\tau_{s'}}\ ds'$$

- It does not depend on coordinates on Riemannian manifold M, i.e.

$$|\mathcal{C}|_{\mathcal{M}} = \int_{\mathcal{C}} \sqrt{\tau_s^T g(s)\tau_s}\ ds = \int_{\mathcal{C}} \sqrt{\tau_s'^T g(s)\tau_s'}\ ds$$

- The sum rule of integral also shows the additivity of $|\mathcal{C}|$. As basic properties of a Riemannian manifold, the proof of these three last properties is omitted. The readers are referred to [30] for further discussions.

Though the shortest distance is provided by following local $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$'s, the geodesic to realize such distance does not necessarily exist due to physical constraints in $\mathbb{E}^3$ space, e.g. the robot cannot tele-transport. Therefore, we need to define a numeric solution to generate the feasible optimal path, which best realizes the proposed concept. To this end, we adopt the k-Nearest-Neighbor (kNN) concept which is mostly applied in manifold learning methods such as Isomap [31]. The projected weights can then be evaluated in a projected space onto the plane supported by $\hat{\boldsymbol{e}}_{\boldsymbol{2}}$ and $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$, which are direction vectors in $\mathbb{E}^3$. In the next section, we introduce the typical trajectory planning for a mobile robot on a point-cloud-based representation.

V. Recipe for Processing Point-Cloud

A. Graph construction and distance mapping

A typical application using the proposed model is the trajectory planning on point-clouds. However, the close-form of planning using the given metric is mathematically intractable. Utilizing the fact that the local geodesic is along the direction of local $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$, we propose an accessible approximation shown as algorithm I, which equivalently maintains the major properties of the proposed metric of (4):

$$
\begin{aligned}
|\mathcal{C}|_{\mathcal{M}} &= \int_{\mathcal{C}} \sqrt{\tau_s^T g(s) \tau_s}\ ds \\
&= \int_{\mathcal{C}} \sqrt{\tau_s^T \left(\mathcal{I}_3 + \frac{e^{k\lambda_1(s)}}{e^{k\lambda_1(s)} + e^{-k\lambda_1(s)}} \hat{\mathbf{e}}_1(s)\hat{\mathbf{e}}_1(s)^T + \frac{e^{-k\lambda_1(s)}}{e^{k\lambda_1(s)} + e^{-k\lambda_1(s)}} \left(\hat{\mathbf{e}}_1(s)\hat{\mathbf{e}}_1(s)^T + \hat{\mathbf{e}}_2(s)\hat{\mathbf{e}}_2(s)^T \right) \right) \tau_s}\ ds \qquad (4) \\
&= \int_{\mathcal{C}} \sqrt{\tau^T \tau_s + |\tau_s^T \hat{\mathbf{e}}_1|^2 + |\tau_s^T \hat{\mathbf{e}}_2|^2 \left(\frac{1}{e^{2k\lambda_1(s)} + 1} \right)}\ ds
\end{aligned}
$$

Algorithm I
ALGORITHM OF TRAJECTORY PLANNING ON POINT-CLOUD

```
1  Create kNN graph $\mathcal{G}(\mathcal{V}, \mathcal{D})$, where the nodes are positions of
   points $\boldsymbol{s_i} \in \mathbb{E}^3$
2  The edges $\mathcal{D}$ is calculated by embedding of Riemannian metric
   in Euclidean space, using the fact that a direction along $\hat{\mathbf{e}}_3$
   is preferred. For site $\boldsymbol{s_i}$ and its neighbor $\boldsymbol{s_i^j}$, distance $d_i^j$ is
   defined as:

   $d_i^j = |\boldsymbol{t_i^j}| e^{1-|\boldsymbol{t_i^j}^T \hat{\mathbf{e}}_{3\,i}|}$, where $\boldsymbol{t_i^j} = \frac{\boldsymbol{s_i^j} - \boldsymbol{s_i}}{|\boldsymbol{s_i^j} - \boldsymbol{s_i}|}$      (5)

3  For each query that consists of a pair of starting and ending
   points, Dijkstra [32] search is implement on $\mathcal{G}(\mathcal{V}, \mathcal{D})$.
```

Considering that $g(s)$ is built in canonical form, following the discussion in section IV.C, the optimal direction is along the direction of $\hat{\mathbf{e}}_3$. By using equation (5), the Euclidean distance is used in this case. Furthermore, weights among sites are defined in a way, such that the diversity between the directions of the path and $\hat{\mathbf{e}}_3$ is exponentially punished.

B. Metrics for evaluation

In order to validate or compare various algorithms for trajectory planning, the following metrics can be used.

- *Number of site visits*: It indicates the number of intermediate points along the path. It can also be interpreted as the shortest distance by considering all weights among sites are unity.
- *Length of trajectory*: The accumulated Euclidean distance of the calculated path.
- *Mean-curvature (MC)*: It is the average of the principal curvatures at a point, i.e. $MC = \frac{\kappa_1 + \kappa_2}{2}$. It reflects to which extent the local neighborhood can be considered as a minimal surface. Intuitively, it represents the local *flatness*. The smaller mean of MC along the path indicates that the path lies on a flatter terrain.
- *Gaussian-curvature (GC)*: It is the product of the two principal curvatures at a point, i.e. $GC = \kappa_1 \kappa_2$. Formally, it depends only on the Riemannian metric of the surface. It represents the local shape change around a point. The smaller mean of GC suggests less *shake* on the path.

VI. Simulation

In this section we evaluate three related methods on simulated dataset. Each simulated point-cloud is uniformly sampled from a parametric surface. The three methods are all based on k-NN graph of point sites, but with different weight definitions.

A. Complex surface

1) Representations: In order to validate the proposed method in a general case, we evaluate the related methods on the point-cloud shown in figure 3. It is sampled from a latent surface, which is arbitrarily defined. The advantage of a parametric surface is that GC and MC can be easily derived [27].

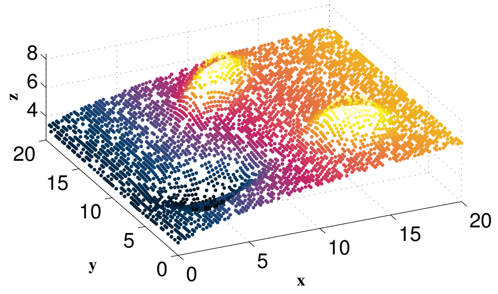

Fig. 3. 4000 points sampled from the surface.

2) Rational path planning: Inspired by the definition of metric tensor, we consider the trace of the Riemannian metric g as the local measure of feasibility for a rational path. Greater trace of g implies a longer incremental curve length, which further induces more divergence of surface orientation and curvature. Figure 4 shows the trace of local metric tensor. Intuitively, the bump areas are equipped with metric of higher trace, which are places that hard to traverse. By arbitrarily given starting and end point beyond bumps, the calculated trajectories are shown in figure 5. The primary observation is that the proposed method (in white) leads to flat trajectory by actively avoiding the bumps in reasonable fashion, though with relatively longer path. A summary of the variations in z (elevation) direction along the path is shown in figure 6. It can be inferred that the proposed trajectory requires less adaptation to the terrain, which leads to easier motion or

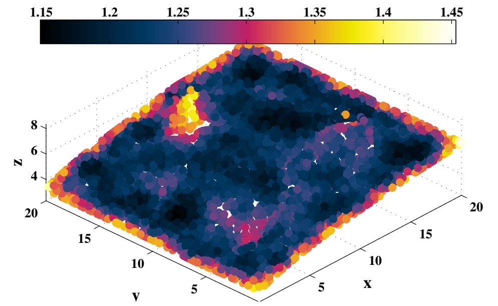

Fig. 4. Trace of the proposed Riemannian Metric. It indicates the roughness of the estimated neighbourhood surface, highlighted by color.

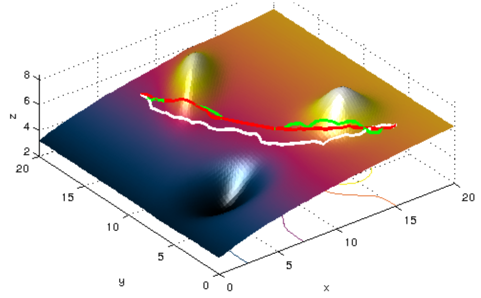

Fig. 5. Generated path for different methods. Green: nearest-neighbour search (21.42); Red: shortest Euclidean geodesic (20.41); White:proposed method (21.90). The numbers in parentheses indicate the total length of the path.

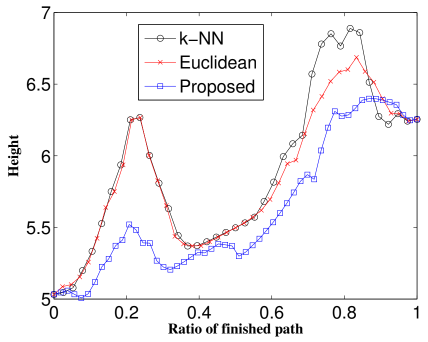

Fig. 6. Height variation along the path for different methods

morphological planning for the mobile robot. Regarding the energy consumption, it can also benefit from the less variation.

3) Validation by the direction of $\hat{e}_3$: The property of the proposed method by Algorithm I is that it is supposed to optimize the path by following local $\hat{e}_3$ direction as much as possible. It is because $\hat{e}_3$ directs the direction of local geodesic, which leads to the most terrain similarity. A qualitative result is shown in figure 7. The $\hat{e}_{3i}$ directions are represented by short lines, and the color definition of paths follows figure 5. We can see that the yellow trajectory calculated by Algorithm I follows mostly the short lines. A

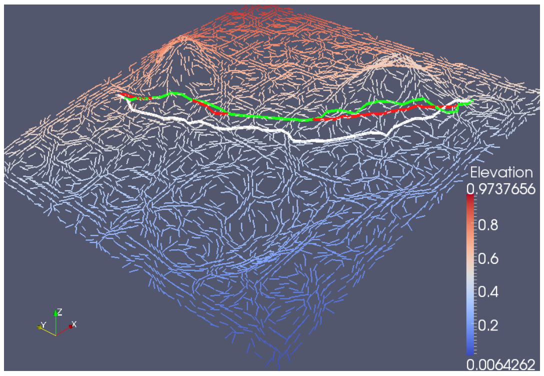

Fig. 7. Comparison to the direction of local $\hat{e}_3$.

quantitative comparison is depicted in figure 8. It shows the statistics of the absolute cosine values of the path separation angle to local $\hat{e}_3$ along the path. A higher value indicates the direction is more similar to $\hat{e}_3$. It indicates that the proposed method fulfills the most coincidence aligning with $\hat{e}_3$.

Method	k-NN	Shortest Euclidean	Proposed
# Nodes	39	43	47
Length	21.42	20.41	21.90
mean MC	0.0946	0.0853	0.0578
std-dev MC	0.1208	0.0989	0.0636
mean GC	-0.0249	-0.0254	-0.0089
std-dev GC	0.0246	0.0232	0.0116

Table I
STATISTICS OF THE TEST RUN. MC INDICATES MEAN CURVATURE; GC INDICATES GAUSSIAN CURVATURE.

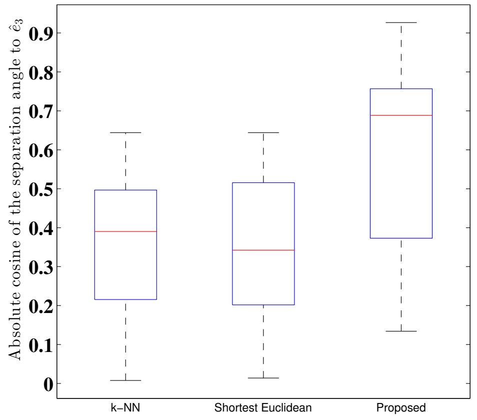

Fig. 8. Boxplot to show the divergence to $\hat{e}_{3i}$ directions along the path.

4) Curvature statistics: Recalling the metrics defined in section V.C, the comparison of curvature dynamics is shown in table I, where the outperforming value is highlighted by gray background. Despite of the longer traveled length, the proposed method holds the minimum MC and GC, as they are the main goal of the design.

VII. TESTS ON REAL DATA

As a sample of use-case, we utilize one of the dataset introduced by [33] for the test on real data. The scenario is shown in top-right of figure 9. Notice that the yellow eclipse marks a fireweed area, which may cause trouble for the robot.

In order to reduce the cost of planning, we sample 10K points out of the raw point-cloud (around 504K points). Adopting the GPU implementation of TVF introduced in [16], the computation time for sparse voting is 24.79ms. Based on Algorithm I, the calculated path is shown in figure 9. It can be observed that the cyan trajectory, by the proposed method, actively avoids the fireweed area, leading to a more rational path for the mobile robot.

The estimation of curvatures on point-cloud is a subtle problem. The readers are referred to [34] for curvature estimation by TVF. The evaluation in terms of GC and MC is omitted.

VIII. CONCLUSION

In this paper, we introduced a Riemannian metric for the representation of point-cloud, which helps the modeling of the

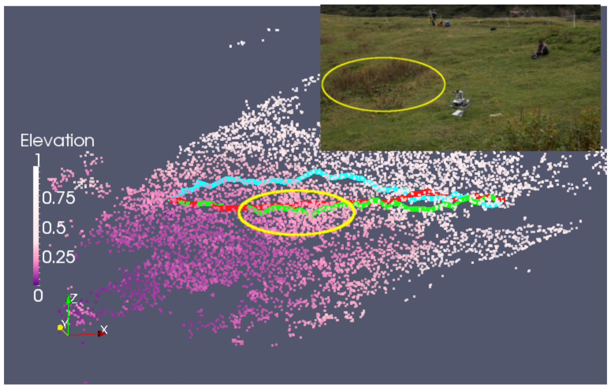

Fig. 9. Scene of the experiment and calculated trajectories for various methods. Green: nearest-neighbour search (24.7m); Red: shortest Euclidean geodesic (22.5m); Cyan: proposed method (29.3m). The numbers in parentheses indicate the total length of the path.

environment, especially aiding the path planning for mobile robots directly on raw point-cloud. Geometrical properties of the proposed metric tensor are discussed, which provide hints for further research related to the modeling problems using raw point-clouds. The proposed framework is validated by path planning task on point-clouds, using both simulated data and real dataset. The results show that the proposed method is able to calculate rational paths for the robots, which facilitate the robotic navigation. Comparison and integration with other robotic tasks are to be carried out in the future.

REFERENCES

[1] G. Pereira, L. Pimenta, A. Fonseca, L. Corrêa, R. Mesquita, L. Chaimowicz, D. De Almeida, and M. Campos, "Robot navigation in multi-terrain outdoor environments," *The International Journal of Robotics Research*, vol. 28, no. 6, pp. 685–700, 2009.

[2] M. Liu, C. Pradalier, and R. Siegwart, "Visual homing from scale with an uncalibrated omnidirectional camera," *Robotics, IEEE Transactions on*, vol. 29, no. 6, pp. 1353–1365, Dec 2013.

[3] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 108–117, 2006.

[4] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[5] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the National conference on Artificial Intelligence*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002, pp. 593–598.

[6] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2D mapping," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 22–29.

[7] R. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–4.

[8] M. Liu and R. Siegwart, "Information theory based validation for point-cloud segmentation aided by tensor voting," in *International Conference on Information and Automation (ICIA)*. IEEE, 2013.

[9] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in *the 12th International Symposium on Experimental Robotics (ISER)*, vol. 20, 2010, pp. 22–25.

[10] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM3D mapping outdoor environments," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[11] G.-J. Kruijff, M. Janicek, S. Keshavdas, B. Larochelle, H. Zender, N. Smets, T. Mioch, M. Neerincx, J. van Diggelen, F. Colas, M. Liu, F. Pomerleau, R. Siegwart, V. Hlavac, T. Svoboda, T. Petricke, M. Reinstein, K. Zimmerman, F. Pirri, and M. Gianni, "Experience in System Design for Human-Robot Teaming in Urban Search & Rescue," in *Field and Service Robotics*. Springer, 2012.

[12] M. Gianni, P. Papadakis, F. Pirri, M. Liu, F. Pomerleau, F. Colas, K. Zimmermann, T. Svoboda, T. Petricek, G. Kruijff *et al.*, "A unified framework for planning and execution-monitoring of mobile robots," in *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[13] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *ACM Siggraph Computer Graphics*, vol. 21, no. 4. ACM, 1987, pp. 163–169.

[14] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 2, 2010.

[15] G. Medioni, M. Lee, and C. Tang, *A computational framework for segmentation and grouping*. Elsevier Science, 2000, vol. 1.

[16] M. Liu, F. Pomerleau, F. Colas, and R. Siegwart, "Normal Estimation for Pointcloud using GPU based Sparse Tensor Voting," in *IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, 2012.

[17] Y. Dumortier, I. Herlin, and A. Ducrot, "4-D Tensor Voting motion segmentation for obstacle detection in autonomous guided vehicle," in *IEEE Intelligent Vehicles Symposium*. IEEE, 2008, pp. 379–384.

[18] J. Jia and C. Tang, "Inference of segmented color and texture description by tensor voting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 771–786, 2004.

[19] H. Schuster, "Segmentation of lidar data using the tensor voting framework," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 35, pp. 1073–1078, 2004.

[20] B. King, "Range data analysis by free-space modeling and tensor voting," Ph.D. dissertation, Rensselaer Polytechnic Institute, 2009.

[21] F. Colas, S. Mahesh, F. Pomerleau, M. Liu, and R. Siegwart, "3d path planning and execution for search and rescue ground robots," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 722–727.

[22] J. Baek, A. Deopurkar, and K. Redfield, "Finding geodesics on surfaces," Technical Report,(November 2007), Tech. Rep., 2007.

[23] R. Kimmel, A. Amir, and A. M. Bruckstein, "Finding shortest paths on surfaces using level sets propagation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 17, no. 6, pp. 635–640, 1995.

[24] M. R. Ruggeri, T. Darom, D. Saupe, and N. Kiryati, "Approximating geodesics on point set surfaces," in *Proc. of the symposium on point-based graphics*, 2006, pp. 85–93.

[25] F. Mémoli and G. Sapiro, "Distance Functions and Geodesics on Submanifolds of R^d and Point Clouds," *SIAM Journal on Applied Mathematics*, vol. 65, no. 4, pp. 1227–1260, 2005.

[26] P. Mordohai and G. Medioni, "Tensor voting: a perceptual organization approach to computer vision and machine learning," *Synthesis Lectures on Image, Video, and Multimedia Processing*, vol. 2, no. 1, pp. 1–136, 2006.

[27] J. Gallier, *Geometric methods and applications: for computer science and engineering*. Springer, 2011, vol. 38.

[28] H. Koo and N. Cho, "Graph cuts using a Riemannian metric induced by tensor voting," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 514–520.

[29] Y. Boykov and V. Kolmogorov, "Computing geodesics and minimal surfaces via graph cuts," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 26–33.

[30] H.M.Khudaverdian, "Riemannian geometry." [Online]. Available: http://www.maths.manchester.ac.uk/~khudian/Teaching/Geometry/GeomRim11/riemgeom11.pdf

[31] M. Balasubramanian and E. Schwartz, "The isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, pp. 7–7, 2002.

[32] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[33] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms," *The International Journal of Robotics Research*, 2012.

[34] W.-S. Tong and C.-K. Tang, "Robust estimation of adaptive tensors of curvature by tensor voting," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 3, pp. 434–449, 2005.